

# Claims

- [c1] 1. A method for dynamically generating program code adding behavior to a program based on attributes, the method comprising:  
adding a component object to a program class of the program to create a component;  
defining at least one attribute specifying declaratively behavior to be added to the program;  
associating said at least one attribute with the component; and  
in response to instantiation of the component at run-time, generating a subclass based on the program class and said at least one attribute, the subclass including dynamically generated program code based on said at least one attribute.
- [c2] 2. The method of claim 1, wherein said defining step includes defining a particular attribute using active metadata, so as to provide a mechanism for generation of program code from said particular attribute.
- [c3] 3. The method of claim 2, wherein said active metadata dynamically generates code for inclusion in a subclass based on the program class.

- [c4] 4. The method of claim 1, wherein the generating step includes generating a subclass comprising an instance of a declared component class.
- [c5] 5. The method of claim 1, wherein the generating step includes generating a subclass comprising an instance of an abstract component class.
- [c6] 6. The method of claim 1, wherein said defining step includes defining at least one attribute based on comments in source code of the program class.
- [c7] 7. The method of claim 6, further comprising: precompiling a class containing static attributes from said comments.
- [c8] 8. The method of claim 7, further comprising: loading the class containing static attributes before subclass generation when a component is instantiated.
- [c9] 9. The method of claim 6, further comprising: defining an automated mapping between attribute syntax in comments and attribute syntax as expressed in generated program code.
- [c10] 10. The method of claim 1, wherein said defining step includes defining at least one attribute in a property file external to the program class.

- [c11] 11. The method of claim 10, further comprising:  
compiling a class containing dynamic attributes from  
said property file.
- [c12] 12. The method of claim 11, further comprising:  
loading the class containing dynamic attributes before  
subclass generation when a component is instantiated.
- [c13] 13. The method of claim 10, further comprising:  
defining an automated mapping between attribute syn-  
tax in a property file and attribute syntax as expressed  
in generated program code.
- [c14] 14. The method of claim 10, wherein attributes in the  
property file comprise property name and property value  
pairs.
- [c15] 15. The method of claim 1, wherein said defining step  
includes defining attributes for a superclass from which  
the program class inherits.
- [c16] 16. The method of claim 1, wherein said defining step  
includes defining attributes for the program class' pack-  
age from which the program class inherits.
- [c17] 17. The method of claim 1, wherein said defining step  
includes defining attributes for an interface from which  
the program class inherits.

- [c18] 18. The method of claim 17, wherein said generating step includes generating an instance of a subclass to mock the behavior of the interface.
- [c19] 19. The method of claim 1, wherein said generating step includes generating code for a non-abstract method body based on an attribute defined for an abstract method.
- [c20] 20. The method of claim 1, wherein said generating step includes generating program code based on comments in a source file.
- [c21] 21. The method of claim 1, further comprising:  
adding expected calls as instances of anonymous inner classes of the program; and  
applying runtime introspection by a generated subclass to verify a sequence of expected calls.
- [c22] 22. The method of claim 1, wherein the component registers itself with a repository when the component is initially activated.
- [c23] 23. The method of claim 22, wherein the repository can be queried to determine components that are active.
- [c24] 24. A computer-readable medium having processor-executable instructions for performing the method of

claim 1.

- [c25] 25. A downloadable set of processor-executable instructions for performing the method of claim 1.
- [c26] 26. A system for dynamically generating program code based on declarative attributes, the system comprising:
  - a component module for creating a component from a program class based on adding a component object to the program class;
  - an attribute module for defining at least one declarative attribute specifying behavior to be added to the program class and associating said at least one attribute with the component; and
  - a module for generating a subclass of the program class in response to instantiation of the component, the subclass including dynamically generated program code based on said at least one declarative attribute.
- [c27] 27. The system of claim 26, wherein the subclass adds tracing behavior to a program.
- [c28] 28. The system of claim 26, wherein the subclass is a subclass of an abstract class.
- [c29] 29. The system of claim 26, wherein said at least one declarative attribute includes active metadata, so as to provide a mechanism for generation of program code.

- [c30] 30. The system of claim 29, wherein said active meta-data dynamically generates code for inclusion in the subclass of the program class.
- [c31] 31. The system of claim 26, wherein the attribute module provides for defining at least one attribute based on comments in source code of the program class.
- [c32] 32. The system of claim 31, further comprising:  
a precompiler for precompiling a class containing static attributes from said comments.
- [c33] 33. The system of claim 32, wherein the module for generating loads the class containing static attributes before subclass generation.
- [c34] 34. The system of claim 31, further comprising:  
an automated mapping between attribute syntax in comments and attribute syntax as expressed in generated program code.
- [c35] 35. The system of claim 26, wherein the attributed module provides for defining at least one attribute in a property file external to the program class.
- [c36] 36. The system of claim 35, further comprising:  
a module for compiling a class containing dynamic attributes from said property file.

- [c37] 37. The system of claim 36, wherein the module for generating loads the class containing dynamic attributes before subclass generation when a component is instantiated.
- [c38] 38. The system of claim 35, further comprising:  
an automated mapping between attribute syntax in a property file and attribute syntax as expressed in generated program code.
- [c39] 39. The system of claim 35, wherein attributes in a property file comprise property name and property value pairs.
- [c40] 40. The system of claim 26, wherein the attribute module provides for defining attributes for a superclass from which the program class inherits.
- [c41] 41. The system of claim 26, wherein the attribute module provides for defining attributes for an interface from which the program class inherits.
- [c42] 42. The system of claim 41, wherein the module for generating generates an instance of a subclass to mock the behavior of the interface.
- [c43] 43. The system of claim 26, wherein the module for generating generates code for a non-abstract system body

based on an attribute defined for an abstract method.

[c44] 44. The system of claim 26, wherein the module for generating includes generating program code based on comments in a source file.

[c45] 45. The system of claim 26, wherein the component registers itself with a repository when the component is initially activated.

[c46] 46. The system of claim 45, wherein the repository can be queried to determine components that are active.

[c47] 47. A method for adding behavior to an application without access to application source code, the method comprising:  
defining at least one attribute specifying declaratively behavior which is desired to be added to an application without access to the application source code;  
storing said at least one attribute in a properties file external to the application;  
creating a dynamic attributes class based on the properties file;  
compiling the application and the dynamic attributes class; and  
generating a subclass which includes dynamically generated code adding behavior to application based on said



at least one attribute.

- [c48] 48. The method of claim 47, wherein said defining step includes defining a particular attribute using active metadata, so as to provide a mechanism for generation of program code from said particular attribute.
- [c49] 49. The method of claim 48, wherein said active meta-data dynamically generates code for inclusion in the subclass.
- [c50] 50. The method of claim 47, wherein said defining step includes defining an attribute for overriding a method of the application.
- [c51] 51. The method of claim 47, wherein said defining step includes defining an attribute for extending a method of the application.
- [c52] 52. The method of claim 47, further comprising:  
loading the class containing dynamic attributes before  
generating the subclass.
- [c53] 53. The method of claim 47, further comprising:  
defining an automated mapping between attribute syntax in the properties file and attribute syntax as expressed in generated program code.
- [c54] 54. The method of claim 47, wherein said at least one

attribute in the properties file comprise property name and property value pairs.

- [c55] 55. The method of claim 47, wherein said creating step includes creating a dynamic attributes class using a pre-compiler.
- [c56] 56. The method of claim 47, wherein said creating step includes creating a dynamic attributes class at runtime.
- [c57] 57. The method of claim 47, wherein said compiling step includes using a Java compiler (JAVAC).
- [c58] 58. The method of claim 47, wherein said generating step includes using a precompiler.
- [c59] 59. The method of claim 47, wherein said generating step includes using a runtime compiler.
- [c60] 60. A computer-readable medium having processor-executable instructions for performing the method of claim 47.
- [c61] 61. A downloadable set of processor-executable instructions for performing the method of claim 47.